R. L. Taylor

Low-End General-Purpose Systems

Since the announcement of the IBM System/3 in 1969, IBM has been incorporating leading-edge technology in products referred to as small general-purpose systems. With the many models of the System/3, System/32, System/34, and System/38, IBM has introduced many technological advances addressing the needs of diverse customers, from the novice, first-time user to the experienced user in the distributed data processing account. By identifying the goals, objectives, design themes, major salient features, and development constraints, this paper reviews and highlights the technical evolution of these products in terms of their systems layout, processor architecture, machine structure, and programming support.

Introduction

The System/3 was announced by IBM in July 1969 and shipped in January 1970. This was the first in what has become a family of four system lines including the System/32, System/34, and System/38. This family has enjoyed significant success in addressing the computing requirements of small general-purpose computer users. A companion paper on IBM's small real-time systems is also included in this issue [1].

These general-purpose systems were originally intended primarily for installations not previously using computers, as was the 1401 [2], and there have been extensive enhancements to these systems in response to expanding customer requirements. These enhancements were affected by improving price/performance technologies which, in conjunction with expanding customer needs, significantly affected the key development decisions.

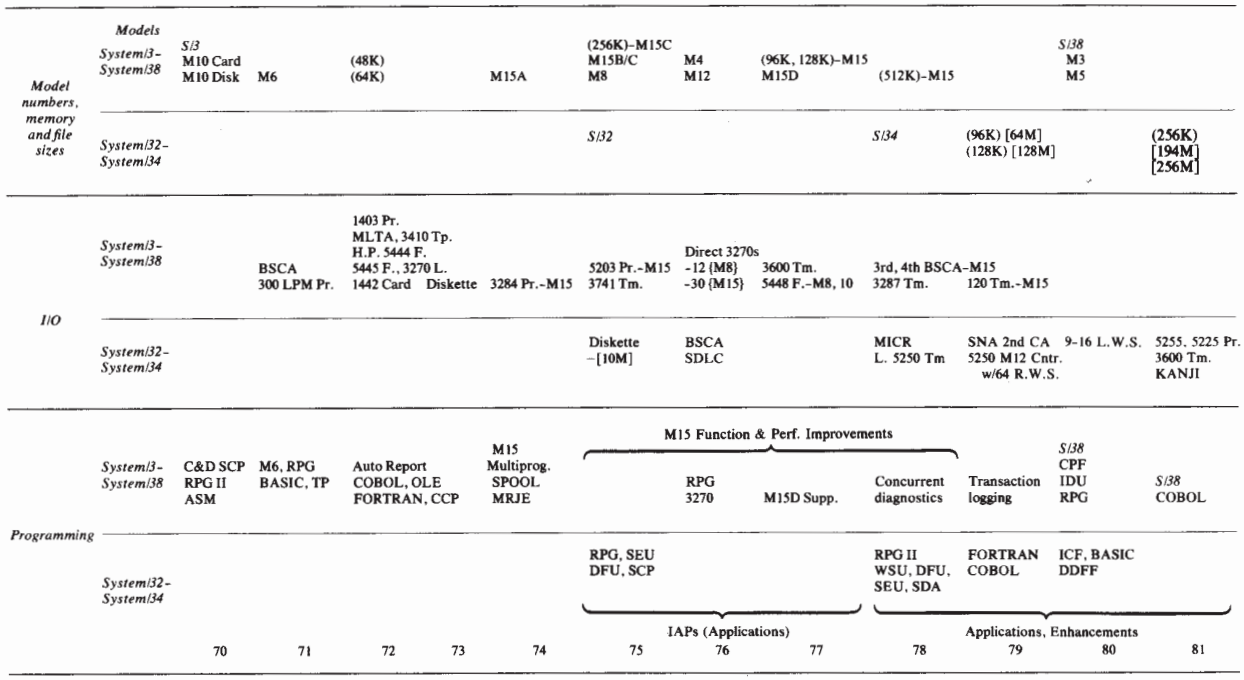
This paper relates the reasons for some of these decisions. It is not intended to be a complete history, but rather to provide some insight as to why certain directions were taken. The following section gives a brief description of technological improvements and key requirements and constraints for each system. The next three sections address the system layout, CPU architecture, and machine structure. This is followed by a section that discusses the evolution of software supporting these products, including operating systems, languages, and applications, and the final section presents a brief summary of the paper and reflects upon future trends.

Small general-purpose systems overview

In 1969, logic technology at IBM had just achieved three logic circuits per chip. In 1978, the System/38 was announced with 704 circuits per chip. In 1969, memory density was less than 100 bits per chip; by 1978, it was 64K bytes per chip (K = 1024) [3]. In file technology, recording densities went from 20 kilobits per square inch in 1969 to 6 megabits per square inch in 1978 [4, 5]. Additional advances in technology have occurred in packaging, power supplies, input and output devices, adapters, and other system components.

These technological improvements have significantly increased capacities. In 1969, a System/3 renting for $2300 per month (without program products) had approximately 3000 circuits in the processor, 24K bytes of main storage, and 10M bytes of file storage (M = 1 048 576). Memory increments were in units of 8K bytes renting for $28.75/K-bytes. In 1980, a System/38 with an equivalent monthly lease charge (again excluding program products) had approximately 20 000 circuits in the processor, 512K bytes of main storage, and 130M bytes of file storage,

Figure 1 Time-line of small general-purpose systems.



All dates refer to the year of first customer shipments.
( ) = memory capacities in bytes; K = 1024
[ ] = file sizes in bytes; M = 1 048 576
ASM = Assembler
BSCA = Binary Synchronous Communications Adapter
C&D SCP = Card & Disk System Control Program(s)
CA = Command Adapter
CCP = Communications Control Program
CPF = Control Program Facility
DDFF = Distributed Disk File Facility
DFU = Data File Utility
IAP = Installed Application Program(s)
ICF = Interactive Communications Feature
IDU = Interactive Data Utilities
LPM = Lines Per Minute
MICR = Magnetic Ink Character Recognition
MLTA = Multi-Line Terminal Adapter
MRJE = Multileaving Remote Job Entry
OLE = On-Line Linkage Editor
SCP = System Control Program
SDA = Screen Design Aid
SDLC = Synchronous Data Link Control
SEU = Source Entry Utility
WSU = Workstation Utility
Cntr. = Controller
F. = File
H.P. = High Performance
L. = Local
Pr. = Printer
R. = Remote
Tm. = Terminals
Tp. = Tape
W.S. = Workstations

with memory increments of 256K bytes available at approximately $.55/K-bytes, or a cost reduction of more than 50 to 1. These advances have allowed corresponding improvements in application justification, customer and IBM support personnel productivity gains, enhanced systems operation and functions, and architectural structure. The evolution of these systems is shown in Fig. 1.

System/3

The initial System/3 was primarily directed toward the new-account customer who was expected to use the system for accounting and financial applications serially executed with transactions batched. A key assumption in selecting interfaces was that the customer would have few trained data processing personnel. The major assumption was the target rental—between $1000 and $2500 per month.

These requirements translated into a number of system objectives. Hardware and maintenance costs had to be substantially reduced compared to those of systems then available. Built-in capacity for on-site functional expansion had to be provided. The user's installation problems had to be minimized by making the system compact and the programming support easy to use. The System/3 Model 10 was IBM's response to these objectives [6]. Based on the new 96-column card and its associated processing machines, a new version of the RPG programming

language called RPG II, new I/O devices including fixed and removable disks, and a new systems architecture, this system was widely accepted as an easy-to-use small batch system. (Cost consideration prohibited use of the System/360 architecture [7].) In March 1971, the support for the System/3 Model 6 offered comparable facilities in a small, versatile, operator-oriented system available without card equipment for batch processing.

The product characteristics of the System/3 resulted in customer application growth and demands for product enhancements. These demands took three major forms. First, support the system as an interactive workstation-oriented system. Second, expand to larger, more diverse configurations with the software to use them efficiently. Finally, provide an interactive operator-oriented system with a rental substantially less than $1000 per month. The following paragraphs expand upon each of these.

Through 1970 and 1971, hardware communications features were announced, but the software support was restricted to single-line Binary Synchronous Communications (BSC) [8] support via an RPG feature and assembler-written user control code. This level of support did not lead to the wide utilization of interactive terminals which was being experienced on large systems [8]. Thus, in May 1972, IBM announced the System/3 Communications Control Program (CCP), which supported an on-line network of terminals with function similar to that provided with larger systems. It enabled users at terminals to invoke applications, allowed programs to access a common set of disk files, monitored the execution of several concurrently executing tasks, and was tailored to suit both batch and on-line environments. This feature provided the facilities necessary to make the System/3 a strong, competitive, interactive system.

Extensive interactive use with CCP and a larger number of transactions processed by a growing number of batch applications resulted in the second major expansion requirement. As Fig. 1 illustrates, this was addressed through a number of extensions/additions of memory capacities, file sizes, magnetic tape, 80-column card equipment, additional printer options, etc. In July 1973, the third System/3 model was announced, the Model 15. This model provided SCP support for multiprogramming, including multiple batch job streams, spooling, device-independent data management, system history area, and a new operator interface. The system and its software, in conjunction with CCP, have provided the base for a multiprogramming, multitasking system which has had many enhancements. There were also three additional versions of this machine model and three new smaller interactive models (Models 4, 8, and 12).

- *System/32*

The third major expansion requirement was for an operator-oriented system leasing for substantially less than $1000 per month. IBM's response was the announcement of the System/32 in January 1975. This was a full-scale, fully-integrated computing system, the size of a desk, for the price of adding a clerk. It incorporated substantial usability features, powerful procedure capability, and extensive industry-oriented applications, allowing installation and use without customer programmers.

- *System/34*

Demand for a lower-cost, interactive, multiple-workstation system with System/32-usability characteristics led to IBM's announcement of the System/34 in April 1977. The System/34 was a powerful, low-cost system available for both batch and interactive environments. Integrated into the system and its System Support Program (SSP), the multitasking workstation support made the System/34 an effective solution for the new-account user, an existing user migrating to the new product, and the larger account employing distributed data processing.

Satisfying these objectives required many features not commonly associated with products in this price range. Examples included multiple processors, including a processor executing system control code; new memory-management functions; microprocessors for device attachment; productivity features; usability and installability improvements; flexible scheduling of work; multinational language support; and broad communications support under the Interactive Communications Feature (ICF). As shown in Fig. 1, the growth in functional capability and options on this system has paralleled that of the System/3.

- *System/38*

In July 1980, IBM shipped to customers the System/3 successor—the System/38. Employing the latest advances in LSI logic circuitry, MOSFET main storage, RAM MOSFET and bipolar control store, system organization, and processor structure, this system supported a new approach to systems design. It provided a high level of function, integrated data base facilities, excellent usability characteristics, highly productive application and systems programmer facilities, and nondisruptive growth.

Examples of unique system functions included flexible facilities for the scheduling of work in the system, comprehensive security and authorization support, powerful message handling facilities, and a unique single-level-store concept. The integrated data base facilities provided described data, simultaneous use of shared data files, integrity facilities, full data save and restore functions, and program independence from stored file structures.

Major usability characteristics included a new control language, a data-description language for defining database-file and display-screen formats, a powerful command-selection menu, and a prompting facility. Enhancements in the installation of new applications were addressed through simplified programming facilities. Included were a new RPG III programming language, on-line program debugging, powerful interactive source and data entry utilities, and a query utility.

Nondisruptive growth was enhanced by the architectural approach. This architecture was structured in a series of horizontal layers, each providing a consistent, uniform, well-architected interface to the implementation above it, allowing replacement through a given layer with minimal impact. Subsequent sections of this paper analyze these systems in terms of their component parts. It is important to remember that in each component area decisions were made on a total system basis. There was no attempt to individually optimize any component.

System organization and I/O device attachment

Advances in technology have had a significant impact on the selection of I/O and its attachment. The I/O characteristics have, in turn, had a major impact on the architecture of the other system elements. Examination of these components shows an evolution in terms of

- Unbuffered, program-controlled, native I/O adapters to fully buffered, autonomous channel attachment.
- Fully synchronous data transfer between device and central processor to fully asynchronous transfer between decoupled processor and channel.
- Primary reliance of unit-record input to reliance on display-oriented and industry-oriented terminals.

- *System/3*

One of the most important new elements of the System/3 was its medium—the 96-column card. Although capacity was increased, higher performance was achieved by reducing the mass and size of the card. Reduced product cost was realized through a design requiring smaller and fewer parts. More reliable techniques for cornering, stacking, etc. were provided, and the card I/O equipment was made more compact. Finally, increased capacity through use of 96 columns was achieved.

A second feature, which subsequently became standard on these products, was an embedded, fixed disk using the fixed-block architecture [9]. The 5444 disk storage drive achieved its low-cost, high-reliability, and capacity requirements by providing two 14-inch magnetic coated disks mounted on a common spindle. Because low-cost load dump media such as diskettes were not available, the upper disk was removable in a cartridge to control contamination and facilitate handling. This allowed system and frequently used application code and data to be resident, with remaining programs and data off line.

The importance of product cost influenced the choice of device adapter characteristics. Two modes of data transfer were provided. For most operations, devices transferred data utilizing a noninterrupting multiplex channel via cycle steal (suspension of processor activity while a memory access was made to service I/O). This reduced the main storage needed for interrupt handling to that required to interrogate I/O completion. In situations where device interrupts were required (*e.g.*, communications) or program execution could not continue (*e.g.*, program check), eight prioritized interrupt levels, each with separate registers, were provided.

To allow concurrent operation of a batch job stream with the Serial I/O Channel, telecommunications, or occasional operator inquiry, a Dual Program Feature (DPF) was provided on the Model 10. This feature allowed two programs to be resident in main storage, with control passing from one to the other.

Expansion into the multiprogramming, multitasking environment required more effective task and I/O data transfer mechanisms. This resulted in an evolution ending with the Model 15D including hardwired (personalized at the AND/OR level with many part numbers) microprocessors performing outboard control of displays, files, and console; masking interrupts by level; two-byte channel operation; and four unique interrupt levels [10].

- *System/32*

The System/32 cost reduction objective required a different approach from the System/3 trend toward more powerful I/O handling. The removable file was deleted and a diskette was provided. The display and keyboard were integrated into the base system, which, with the software support providing messages requiring indicators on the System/3, gave enhanced service. To minimize cost, all I/O operations and data transfer were controlled by the central processor [11].

The native instruction set of the System/32 was optimized for effective I/O control. The logic in the device adapters was minimal. The processor performed data transfer, including transfer by keystroke for the keyboard, by byte for the diskette, and by sector for the file. To properly control I/O and error conditions, an interrupt mechanism was provided, consisting of seven interrupt levels, each level having a complete complement of registers.

Denser FET nonlinear load circuits and logic technology and packaging considerations allowed the processor and channel to be packaged on one logic board, and the I/O adapters on single cards.

In attaching the 1255 Magnetic Character Reader, the System/32 began a trend followed by the later systems. This adapter included a programmable microprocessor which had been originally developed for the System/7 [1]. This microprocessor, with a 4-bit data flow, provided for 16-bit instructions and 16-bit addresses. This unique combination of data flow, address, and instruction lengths resulted in a low product cost with efficient bit handling.

- *System/34*

In choosing attachment mechanisms on the System/34 and System/38, programmable microprocessors were used as adapters. This had substantial advantages:

- Increased efficiency in the handling of data transfer and device control.
- Expandability; for example, the movement of code executing in the central processor to the adapter.
- Flexibility, allowing changes during development.
- Development, by minimizing engineering-change cost.
- Terminal independence with the potential of handling device-unique functions in the microcode [12].
- Configurability, potentially providing on-line operation diagnostics and service.

The System/34 employed the same microprocessor used by the System/32 not only for 1255 attachment but also for printer and workstation attachment. With more demanding attachments resulting from more workstations and networking than this microprocessor could contain (*e.g.*, multiline telecommunications), a more powerful controller was required. A choice of a more powerful microprocessor was required. The requirements for such a microprocessor included device-handling functions, buffering, multiple interrupt levels, effective polling functions, reasonable product cost, and a straightforward instruction set for ease in programming. These were nearly the same objectives used in the design of the System/32 processor. A processor with this architecture was already included in the System/34 CPU complex, so the choice was straightforward. A version of this processor was included as the second System/34 controller.

Performance requirements for the high data transfer rates of direct disk access devices required that these devices on the System/3, System/32, and System/34 be attached with hardwired controllers.

- *System/38*

These trends toward a more powerful I/O structure continued with the System/38 [13]. The demands of larger configurations joined with the opportunities of new cost/performance technology to provide a sophisticated system organization. The following objectives were identified:

- Use the system's virtual addresses in the channel.
- Exploit LSI technology.
- Provide multiple I/O attachment interfaces.
- Make the processor and channel asynchronous.
- Allow multiprogramming in the channel program.

This system employed a sophisticated virtual memory addressing scheme. To avoid the problems inherent in channel use of real addresses, such as programming relocating addresses, a Virtual Address Translator interface, separate from that of the processor, was provided. This enabled the channel to execute commands or programs containing virtual addresses, and by saving the translated real addresses, to then cycle steal data into locations identified by these real addresses. Special controls allowed addresses to be retranslated during I/O operations.

The System/38 contained two types of adapters, one hardwired for high-speed devices [14], the other microprogrammed for lower-speed devices [12]. In the latter case, a standard microprogrammed I/O processor had multiple uses. This microprocessor, packaged with the channel adapter as a field-replaceable unit, provided device control function. Separate data store was provided for buffers, control tables, channel queues, and work areas. The microprocessor had an 8-bit data flow, 8-bit data addresses, 13-bit control store addresses, 32 LSRs (Local Storage Registers) in the first 32 storage locations, and two program levels with a single interrupt. This controller was used for unit-record devices, for local workstation support, and for the communications adapter [12]. The other adapter type was hardwired for high-speed magnetic media control. It provided for high-speed data transfer and multiple overlapped time-shared execution of RAM instructions. Although not providing simultaneous data transfer from multiple drives, it did allow seek overlap and rotational sensing identifying which drive was ready. It also provided a standard, consistent data flow supporting the channel interface and functions [14].

The requirements of multiplexing subchannels and interfacing with control programs of the main processor are not unlike those of multitasking system programs. This function was provided in the task dispatcher of the system, implemented as a part of the microcode of the machine. To ensure decoupling of channel and processor operation, this same mechanism was used for channel operation. This had the further advantage of allowing multiprogramming at the channel program level.

Central processing unit (CPU) architecture

Fundamental decisions in CPU design involve the architecture and functional organization. The primary factors influencing both these areas include the cost of logic and memory; the expected "typical" use of the system, including importance of high storage efficiency and functional levels of the instruction set; performance considerations such as the relationship of internal processor clock time to memory cycle; architectural independence of technology choices; ease of implementation (*e.g.*, completeness and symmetry of data types); and extendibility considerations in the operation encoding, size of addressing space, number of registers, nature of data types, etc.

- *System/3*

The key objective was lowest cost on a system basis. The target was execution of commercial programs written using RPG II. Efficiently compiling RPG II source code into machine-readable object code was essential. Relatively high storage cost demanded high storage efficiency. The original system was planned to have models with as much as 24K, but extendibility to 64K was to be allowed. Limited multiprogramming was to be supported. Transaction data had to be exchangeable with other systems.

On the basis of these considerations, the data types chosen were characterized by EBCDIC bytes and included both zoned and packed decimal for RPG II manipulation of data entered from character-oriented devices. Binary format data in two's complement form was supported for interchange as well as for control program efficiency. For storage utilization reasons, the byte was chosen as the addressable unit, with two-byte addresses allowing addressability to 64K. Storage efficiency also caused selection of variable-length instructions of three to six bytes with three operand-addressing formats. Each instruction contained a one-byte operation code and a one-byte extender. For reasons of cost and expected use, no general-purpose registers and only three index registers were provided [10].

The processor was implemented via hardwired Monolithic System Technology (MST) logic utilizing approximately 3000 circuits, yielding a processor with a cycle time of approximately 1.5 μs. The speed of the logic technology allowed the processor to perform multiple subcycles within each cycle. Typical device operation was designed without operation-end interrupts, and multiprogramming support was limited to the Dual Programming Feature. The processor utilized a one-byte data path for both instruction fetch and execution cycles.

With extension into larger configurations and more complex environments, many decisions in the functional organization changed in newer models. The instruction set, for compatibility reasons, remained almost the same. This can be exemplified by the Model 15D.

Through address translation, the maximum main storage size was increased to 512K bytes, ECC (Error Correcting Code) was introduced to provide correction of single-bit errors and detection of double-bit errors, and write/fetch storage protection under program control was provided in 2K-byte segments. The address-translation function used 32 8-bit registers residing between the Storage Address Register and main storage. The processor saved the 11 low-order bits of a logical address, and used the 5 high-order bits to select one of 32 translation registers. The 8 bits from this register were concatenated to the 11 low-order bits to form a 19-bit address, thereby allowing 512K-byte addressability. This accommodated programs up to 64K.

The data path of the processor was further changed, allowing a two-byte data channel and instruction-fetch, a privileged mode of operation, maskable operation-end interrupts from all devices for prioritized task switching, and complete overlap of all I/O without data overrun.

- *System/32*

The System/32 processor, for product cost reasons, had to fulfill a dual role. In addition to executing the stored program, it had to be an efficient device controller. Since the System/3 architecture had been formulated, the cost of storage had been substantially reduced. The objective was to effectively use this reduced storage cost to maximum benefit, and to maintain the System/3 instruction set to reduce development cost and time by reusing portions of the System/3 control code. The hardwired control function in the processor was replaced with writable-control-storage-resident code—microcode, which emulated the System/3 instruction set [11].

The instruction set included register-to-register, register-immediate, and register-storage instructions, with eight 16-bit registers. The unit of addressing was a 16-bit word with 16-bit addresses. Five interrupt levels were used, with each having a full complement of eight registers for high-speed interruptibility ensuring no data overrun when handling device data transfer.

- *System/34*

The objectives for the System/34 processor included improvement in price/performance; sufficient processor capacity to support an integrated multitasking, multiprogramming environment; substantial extendibility; Sys-

tem/32 compatibility to utilize some control code from that system; and addressability beyond 64K.

An effective way of providing additional processing performance is to add more processors. This technique was used to offload I/O control by utilizing microprocessors, resulting in more cycles available for program execution. However, an additional step was required to meet all the objectives: multiple processors for CPU functions. Requirements previously stated made the approach straightforward: nonhomogeneous processors each performing special-purpose functions, one for system control and one for execution of stored programs. The compatibility requirement made the choice of processor architectures obvious: the System/32 processor architecture for the former (referred to as the Control Store Processor or CSP), the System/3 processor for the latter (referred to as the Main Storage Processor or MSP).

Offloading of device control allowed execution of control program functions, such as task dispatching and memory management, in the CSP. This multiprocessing approach required an architectural extension to provide for processor synchronization. For the MSP, a supervisory call function requesting service from the CSP was added. CSP changes included an interrupt level dedicated to MSP service, additional instructions for control of the MSP, and a control-mode register to save status when interrupted by an MSP-executing program.

The requirement to address more than 64K of main store was satisfied with address translation as it had been on the System/3. For the System/34, memory was shared by two processors, so two sets of address-translation registers were provided [15].

- *System/38*

The processor on this system was designed to exploit LSI technology to support the high-level architecture. An example of how this was achieved is provided by the functional organization of the interface to memory and its management [16]. Objectives for memory included:

- Take full advantage of the minimum size to provide the lowest cost while supporting full function.
- Facilitate nondisruptive growth by incremental storage growth with no loss of natural performance.
- Decouple processor and memory technology.
- Utilize efficiently the very dense, low-cost, but relatively low-speed memory.

A microprogrammed executable interface for the functions supporting the high-level machine was highly desirable. However, the main-memory speeds prohibited use of this memory for the control code. Thus performance demands required that control store be decoupled from main store, so 8K 32-bit words of RAM control store were added to the system. However, even this control store had cycle times that were relatively long when compared to the possible internal processor clock cycle. Thus, resources in the data flow were partitioned so that a single microinstruction could simultaneously operate on multiple elements. The interface was supported by 32-bit instructions that employed this parallelism and were directly executed by the hardware. Providing the functions through these relatively wide microinstructions (horizontal microcode) had the further advantage of reducing the size of relatively expensive control store needed.

Extendibility and generality considerations demanded very large addressability at this executable interface. This objective was satisfied by providing a 48-bit virtual address yielding a 281-trillion-byte address space. This was very large when compared to the maximum electronic store available with the system. Thus, a sophisticated address-translation process was developed [17].

Providing the high-level function at the executable interface supported by horizontal microcode was a multilevel queue-driven task-control structure. This included a prioritized task dispatcher integrating I/O and program processing tasks, a queued message-handling facility, stack-manipulation functions, sharing, index handling, and a powerful call/return mechanism.

This internal interface included 16 six-byte registers which were used as six-byte base registers or partitioned into four-byte and two-byte registers. Eight of the registers could be further partitioned into 16 one-byte registers. The scalar data types supported were binary, zoned, and packed decimal data, character data, and two- or six-byte address data. The instruction formats were very similar to those employed by System/360–370.

High-level machine structure

The System/32 processor executed instructions for I/O control and instruction-set emulation. The addressable storage for each function was separate from the other. During development, it became clear that the memory for the nucleus of the operating system and the microcode executing in control store would both be exceeded. This would have required two additional storage increments, a major product cost increase. However, by recoding some portions of the nucleus in control-store code, only a single increment was required.

With the addition of a second processor, the offloading of I/O control to microprocessors, and the decreased cost of storage, the System/34 was implemented with more

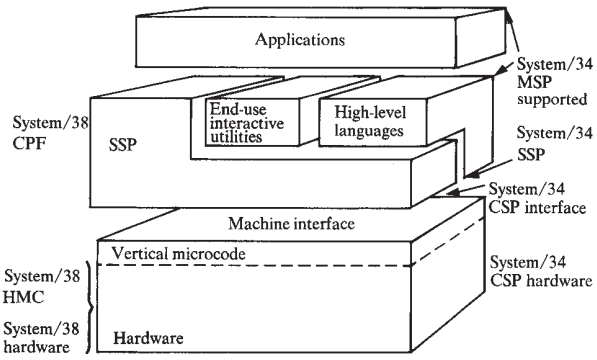


Figure 2 High-level system architecture. Legend: CPF—Control Program Facility; HMC—Horizontal Microcode; SSP—System Support Program; MSP—Main Storage Processor; CSP—Control Store Processor.

system-control code written in microcode executed by the CSP. This included supervisor-call handling, I/O control functions, memory management, transient handling, and task dispatching. As it was enhanced, the System/34 experienced another benefit. Movement of this code to the CSP effectively made the MSP instruction set more functional and allowed enhancements and algorithmic replacement of CSP code without change to MSP code.

The System/38 formalized this concept [18]. As shown in Fig. 2 for both System/34 and System/38, the system structure was based on a set of horizontal layers, each presenting an architected opaque interface to the next layer. The hardware interface was provided through a set of instructions to the horizontal microcode, which interfaced through the executable interface to the vertical microcode, which interfaced to the program products and applications through an architected Machine Interface (MI). Executing through the machine interface, the System Support Program, called the Control Program Facility (CPF), interfaced to the other program products through a controlled set of macros, and to the customer applications and operations through the Command Language (CL) and the Data Description Specifications (DDS). The focus of this section is the vertical microcode supporting the architected Machine Interface (MI). The key objectives addressed by this interface included

- Increased program and data independence from hardware implementation and configuration.
- Increased consideration of RAS.
- Extended integrity functions in the machine.
- Machine-supported security and authorization.
- Efficiently supported, commonly used functions.
- Supported key system functions.

This resulted in a machine interface functionally similar to a typical high-level language definition. System independence was achieved by absorbing hardware dependencies into vertical microcode. Examples included addressing all elements as if they were a part of a single-level store, and relying on the microcode to handle store allocation and transfers between levels of store; no architected bit representation for objects; hidden internal data structures and relationships; and generic instructions late-bound with respect to data type and length.

The MI enhanced RAS considerations [19] in several ways. Every function had a well-defined, consistent, architected interface for which there was completeness, symmetry, and mapping among all data types, and for which there was only one way of performing each function. Internally there were standard synchronous and asynchronous mechanisms. Extensive machine service functions were available to diagnose, isolate, and correct problems. Approaches conducive to providing reliability in code written against that interface (*e.g.*, standard machine-supported call/return mechanisms) were provided.

The MI improved system integrity characteristics through an object-oriented interface in which each construct (program data object, control block, work space, etc.) was explicitly created with certain capabilities defined in the object and checked prior to use. Addressing of any object was via a symbolic pointer which the machine resolved and which, if modified, could no longer be used for addressing. The machine also assumed responsibility for the management of user-oriented data, providing facilities for the backup, retrieval, and maintenance of the data.

Advances were provided in security and authorization through MI. By ensuring process execution independence, each user was isolated from others. All resources were owned through an object called a user profile which determined the user's protection domain. All objects were referenced through capability addressing.

The MI provided efficient support for commonly used program functions. For example, late binding with respect to data type, length, and location for computational instructions was provided by dictionary addressing, referencing values within a space object associated with the code in that program. Powerful generic instructions were provided. Complex data types such as array processing were supported.

Efficient execution in the multitasking, multiprogramming environment required machine provision for control functions. The MI contained either total sys-

tem support or substantial facilities, particularly in the execution phases (as opposed to creation or maintenance) for management of storage, processor allocation, events and exceptions, source/sink device and data handling, data base, communications, and resource management.

One way of characterizing a system is to examine its concurrency attributes. Most systems have provided for some level of asynchronous machine operation, minimally I/O transfer. The MI substantially increased the number and changed the nature of asynchronously executing facilities. These included dispatching functions, storage management functions such as memory and DASD page reclamation and directory management, object creation, system recovery functions, save/restore operations, and asynchronous linkage facilities. Synchronous operation was maintained in the scan, decode, and execution of computational and control instructions, security and authorization checking, and exception handling.

Usability and productivity considerations were primarily directed to the external user interface. The intent was to assume as much management of resources by the system as possible. Machine independence considerations, in turn, required that these resources be managed by the machine. These requirements had to be balanced against performance considerations, requiring at least partial resource management by MI processes. Care was exercised in such instances to ensure that these resource management functions were not tied to specific choices of application structure, functions, or attributes. This allowed performance tuning and configuration changes to be made without affecting the application. Further, user involvement was minimized by containing these functions within the CPF.

The following are examples of special provisions for performance. MI control of resource management constructs was provided, allowing a dispatching priority to be specified for a process. Main store could be pooled, so a specific set of processes was limited in the number of pages used for its working store. Pages could be grouped by affinity to allow access by group. Multirecord sequential data base transfer could be specified, and was particularly useful for performing functions such as sorting. Integrity of address pointers was ensured through use of special memory tags set if a pointer was modified, thereby disallowing use of that pointer for addressing. Alternatively, objects containing pointers could have been required to be in segments separate from data. Finally, the overhead considerations in the interpretation of the generic late-bound high-level instruction set resulted in a step to prebind an MI program.

This concept of an architected, high-level machine interface was the logical evolution of the increased capabilities of the system hardware. It represented an effective way to address major design issues for the entire system. As the largest and latest of these products, the System/38 represented a continuation of a trend that began with the introduction of the System/32.

Operating systems/languages

Since productivity dictates data processing personnel costs, the software support has substantial leverage on the customers' investment. The key decisions made relative to the software concerned 1) its operational effectiveness, 2) the choice of system environments and application structures, 3) the technical concepts employed in the design, 4) the nature of interfaces, 5) the distribution of system support, and 6) IBM-supplied applications.

Operational effectiveness refers to two factors. First are the human factors provided for each user. The second refers to the system performance attributes measured both in terms of batch and interactive transaction rate and interactive response time. As technology costs allowed addressing of new environments and higher transaction rates, additional system functions and interfaces were provided. The programming support increased in generality through, for instance, remote/local transparency for displays. It provided the new 5250 terminal family support with enhanced operator guidance such as menu selections, levels of system messages, etc. It supplied more powerful functional capabilities, including workstation application tasks which were interruptible. Additionally provided were increased recovery facilities such as automatic system recovery following power failures, and simplified installation with the system operational as distributed.

System environment types include batch, real-time, and interactive. The original System/3 supported only the batch environment. (The systems described in this paper do not address the real-time environment [1].) With the System/3 CCP, the user had available a subsystem supporting general-purpose interactive transactions and limited transaction-driven processing. The System/34 generalized this through inclusion of the necessary support in the System Support Program [15]. The System/38 continued this trend by providing full functional capability for general-purpose interactive transactions and by enhancing transaction-driven applications through a transaction-routing facility with a control language interface. It also addressed time-sharing with such interactive productivity aids as the ability to dynamically set checkpoints or display variables interactively.

Technical concepts include a number of topics: the means by which a system binds data to programs, data and programs to store, etc.; units of and algorithms allocating resources; asynchronism of task execution dependent on sharing; the means of handling data; and concurrency attributes as defined by the degree to which the system supports multitasking and multiprogramming. Space limitations prohibit a detailed examination of the evolution within these systems of each of these attributes. Concurrency will, however, serve as a typical example of the increasing capability in these systems over time.

System/3 allowed for single-job-stream execution with all system resources not required by the supervisor available to each program. With CCP, a subsystem executing under control of SCP for such functions as program load, I/O, console management, and physical communications, up to 15 multiple predefined tasks concurrently resident and executing could share a limited number of predefined system resources including data files and terminals.

The Model 15D expanded concurrency by supporting three independent partitions, multiprogrammed with SPOOL, to which serially reusable resources could be allocated. Memory was allocated during system generation allowing later modification. The user could prioritize the scheduling of jobs and dispatching of tasks [6].

Generalization continued with the System/34 SSP. The interactive support was integrated. No absolute constraint was provided on the number of tasks concurrently executing. The SSP dynamically managed prioritized balancing of workstation response characteristics with batch throughput. A single workstation could initiate multiple tasks and a single task could perform at multiple workstations. Program swapping between the file and memory allowed storage overcommitment. Diagnostics could run concurrently with user tasks [15].

This increase in flexibility continued in the System/38 [20]. Resource management functions were implemented in the machine with all objects contained in a single level of storage. The requirement to support a wide variety of interactive transaction types resulted in a new concept, the user-defined subsystem. This allowed the user to tailor one or more operating environments into which his jobs could be grouped, without restricting the application, through a single rule-driven mechanism [16].

The interfaces to a system include 1) the traditional high-level languages specifying the algorithms in the application; 2) specification languages defining the configuration, screen formats, record formats, etc.; 3) utility functions; and 4) languages for operator and job control.

Key to understanding the interfaces on these systems is the role played by RPG. The most commonly used programming languages in the 1960s were COBOL, FORTRAN, BASIC, and PL/I. These languages are all procedural, requiring the programmer to think through his execution sequence and explicitly control files, labels, data structures, etc. This meant that the programmer's time had to be divided between the application and the system. It also meant that additional effort was required to document and debug programs. These attributes were judged to be unacceptable for the users of these systems, even in light of their corresponding advantages, which included complete control and support for all forms of I/O and the ability to optimize on resources. Thus, the decision was made to provide these systems with RPG as the primary language.

RPG was defined in the 1950s to execute on the IBM 1401, providing easy preparation of business reports from record descriptions. It was a nonprocedural language executing with fixed-flow logic allowing the programmer to focus on his file and record descriptions and data operations, not on system characteristics, resources, housekeeping, and documentation. A new version was defined and implemented on the System/3—RPG II.

As these systems addressed larger configurations, more complex applications, and different operating modes, two directions were taken. First, other high-level languages were provided on some systems (FORTRAN, COBOL, and BASIC are supported). Second, the disadvantages inherent in RPG as a nonprocedural language were addressed through additional extensions. A new language version, RPG III [21], was provided on the System/38. This version enhanced execution sequence control for I/O and data base processing, data and device independence, structured programming, and variable binding of any program element through prespecification.

RPG was primarily oriented toward the batch processing of application data. With the emphasis for on-line interactive applications on the System/34, a new tool for program development was produced—the System/34 Work Station Utility (WSU). WSU was a nonprocedural, fixed-format application language based on RPG. It was intended for fast transition to workstation applications performing interactive entry, edit, and correction.

The second class of interfaces consisted of the specification languages. System/3 required that each program contain its own version of the environment: its device descriptions, record formats, file formats, and screen descriptions. Evolving use required a new set of principles emphasizing modularity in application construction through the separation of definitions from logic flow.

Various tools were provided, including screen definition via the System/3 Display Format Facility, device assignment via the control language, etc. On the System/38, the Data Description Specification [22] provided the interface for both data base physical descriptions and logical usage [23] and screen format definition. In addition to standard system-utility functions, the increasingly interactive orientation of these systems required inclusion of display-oriented interfaces, including

- Nonprogrammer handling of data files.
- Source entry, including syntax checking.
- Screen design for easily defining, updating, and displaying screen formats.
- Inquiry and query facilities to display information.

The final interface, the control language, controlled system operation. System/3 employed a single syntax supporting its two control language functions: The Operation Control Language (OCL) used in batch processing provided the basis for job streams, and the Operation Control Commands (OCC) allowed operator control from the system console. As the system expanded into new modes of operation, additional interfaces were provided. For example, CCP had two additional interfaces for its generation and for the assignment of specific sets of terminals, files, programs, and system environment.

The System/32 and the System/34 incorporated their interactive support as part of the base product. These functions were included within more powerful implementations of OCL and OCC, providing parameter passing, substitutions, conditional statements, and nesting.

The System/38 continued this integration, including within its control language interfaces to its significant new functions. However, the need for a flexible control language allowing system function to be invoked from a program, the system console, and a remote or local terminal with user tailorability demanded a new syntax. The System/38 control language used a free-form syntax with commands allowing keyword or positional parameters [24]. Its implementation was based on a rule-driven approach in which a detailed description of each command was stored in a rule describing validity checking, defaults, prompt text, etc. This resulted in improved command consistency and reduced development requirements. In addition to command procedures for inclusion in batch job streams, it provided a compiler offering display operation, declared variables, and IF/THEN/ELSE and DO functions.

Another area which experienced growth was the distribution of data processing. Requirements in this area evolved over time from remote job entry utilities with batch transmission to facilities allowing the user to write applications with the system handling all communications protocol and remote system dependencies. The best example of the latter type of support was the System/34 Interactive Communications Feature (ICF) [25]. ICF allowed the user program to interface to a remote system in exactly the same manner it interfaced to a display. ICF handled all device and communication dependencies, provided the communications link and error recovery, supported batch and interactive communications, allowed remote invocation of System/34 programs, and interfaced with CICS/VS, IMS/VS, 3270 BSC emulation to System/370, CCP, and other devices using BSC protocols. In addition, the Distributed Disk File Facility allowed an application program to access a disk file located on another System/34 or System/3 Model 15D.

A critical element in the marketing success of these systems has been the IBM-provided application products. The typical customer, without trained data processing personnel, wanted end-use application solutions. IBM, with the relatively low cost of these systems, needed to reduce marketing expense by increasing field productivity. The solution was to provide industry-oriented and country-specific application products consisting of commercial and financial applications, tailorable to specific requirements, allowing service for each user's specific system and skill level. The intent was to provide turnkey solutions to customer application needs.

Summary and conclusions

IBM has provided small general-purpose systems which have employed and initiated advances in technology, systems architecture, and software support to meet customer needs. The original System/3 concept has evolved to the point where, in the System/34, it continues to provide a solid architectural base for future expansion. A revolutionary new approach to systems design and architecture in the System/38 has been used to provide unique capabilities to its users. These systems have demonstrated how the advantages of lower-cost, better-performing technologies were used to provide expanded system capabilities meeting ever-growing customer needs.

These trends can be expected to continue. Technological developments and customer requirements will continue to allow, if not demand, continued expansion of the product line. This expansion will occur in three dimensions: in terms of cost, both upward and downward; in function, by addressing new, advanced application areas of data processing, including support of new industry-oriented terminals; and in continued improvement of price/performance. Continuing improvements and expan-

sions of distributed data processing support can be expected. Further technological advances will be applied to continue to reduce the complexity of using these systems. These changes can be expected to be both evolutionary and revolutionary, achieving new system breakthroughs.

References

1. Thomas J. Harrison, Bruce W. Landeck, and Hal K. St. Clair, "Evolution of Small Real-Time IBM Computer Systems," *IBM J. Res. Develop.* **25,** 441–451 (1981, this issue).
2. C. J. Bashe, W. Buchholz, G. V. Hawkins, J. J. Ingram, and N. Rochester, "The Architecture of IBM's Early Computers," *IBM J. Res. Develop.* **25,** 363–375 (1981, this issue).
3. E. W. Pugh, D. L. Critchlow, R. A. Henle, and L. A. Russell, "Solid State Memory Development in IBM," *IBM J. Res. Develop.* **25,** 585–602 (1981, this issue).
4. L. D. Stevens, "The Evolution of Magnetic Storage," *IBM J. Res. Develop.* **25,** 663–675 (1981, this issue).
5. Andrew A. Gaudet and Bobby J. Smith, "An Overview of the Technology in the IBM 3370 Direct Access Storage," *Disk Storage Technology*, 1–2 (1980); Order No. GA26-1665-0, available through IBM branch offices.
6. *The IBM System/3 Model 15D Cyclopedia*, Order No. Z280-0076, available through IBM branch offices.
7. A. Padegs, "System/360 and Beyond," *IBM J. Res. Develop.* **25,** 377–390 (1981, this issue).
8. David R. Jarema and Edward H. Sussenguth, "IBM Data Communications: A Quarter Century of Evolution and Progress," *IBM J. Res. Develop.* **25,** 391–404 (1981, this issue).
9. David L. Nelson, "The Format of Fixed-Block Architecture in the IBM 3370 DAS," *Disk Storage Technology*, 34–35 (1980); Order No. GA26-1665-0, available through IBM branch offices.
10. *IBM System/3 Models 8, 10, 12, and 15 Components Reference Manual*, Order No. GA21-9236, available through IBM branch offices.
11. *S/32 Functions Reference*, Order No. GA21-9176, available through IBM branch offices.
12. E. F. Dumstorff, "Application of a Microprocessor for I/O Control," *IBM System/38 Technical Developments*, 28–31 (1978); Order No. G580-0237, available through IBM branch offices.
13. R. L. Hoffman and F. G. Soltis, "Hardware Organization of the System/38," *IBM System/38 Technical Developments*, 19–21 (1978); Order No. G580-0237, available through IBM branch offices.
14. J. W. Froemke, N. N. Heise, and J. J. Pertzborn, "System/38 Magnetic Media Controller," *IBM System/38 Technical Developments*, 41–43 (1978); Order No. G580-0237, available through IBM branch offices.
15. *System/34 System Support Reference*, Order No. SC21-5155, available through IBM branch offices.
16. L. A. Belady, R. P. Parmelee, and C. A. Scalzi, "The IBM History of Memory Management Technology," *IBM J. Res. Develop.* **25,** 491–503 (1981, this issue).
17. M. E. Houdek and G. R. Mitchell, "Translating a Large Virtual Address," *IBM System/38 Technical Developments*, 22–24 (1978); Order No. G580-0237, available through IBM branch offices.
18. S. H. Dahlby, G. G. Henry, D. N. Reynolds, and P. T. Taylor, "System/38—A High-Level Machine," *IBM System/38 Technical Developments*, 47–50 (1978); Order No. G580-0237, available through IBM branch offices.
19. M. Y. Hsiao, W. C. Carter, J. W. Thomas, and W. R. Stringfellow, "Reliability, Availability, and Serviceability of IBM Computer Systems: A Quarter Century of Progress," *IBM J. Res. Develop.* **25,** (1981, this issue).
20. H. T. Norton, R. T. Turner, K. C. Hu, and D. G. Harvey, "System/38 Work Management Concepts," *IBM System/38 Technical Developments*, 81–82 (1978); Order No. G580-0237, available through IBM branch offices.
21. *S/38 RPG III Reference and Programmer's Guide*, Order No. SC21-7725, available through IBM branch offices.
22. C. D. Truxal and S. R. Ridenour, "File and Data Definition Facilities in System/38," *IBM System/38 Technical Developments*, 87–90 (1978); Order No. G580-0237, available through IBM branch offices.
23. W. C. McGee, "Data Base Technology," *IBM J. Res. Develop.* **25,** 505–519 (1981, this issue).
24. J. H. Botterill and W. O. Evans, "The Rule-Driven Control Language in System/38," *IBM System/38 Technical Developments*, 83–86 (1978); Order No. G580-0237, available through IBM branch offices.
25. *System/34 Data Communications Reference*, Order No. SC21-7703, available through IBM branch offices.

Received June 27, 1980; revised November 7, 1980

The author is located at the IBM Information Systems Division laboratory, 3605 Highway 52N, Rochester, Minnesota 55901.